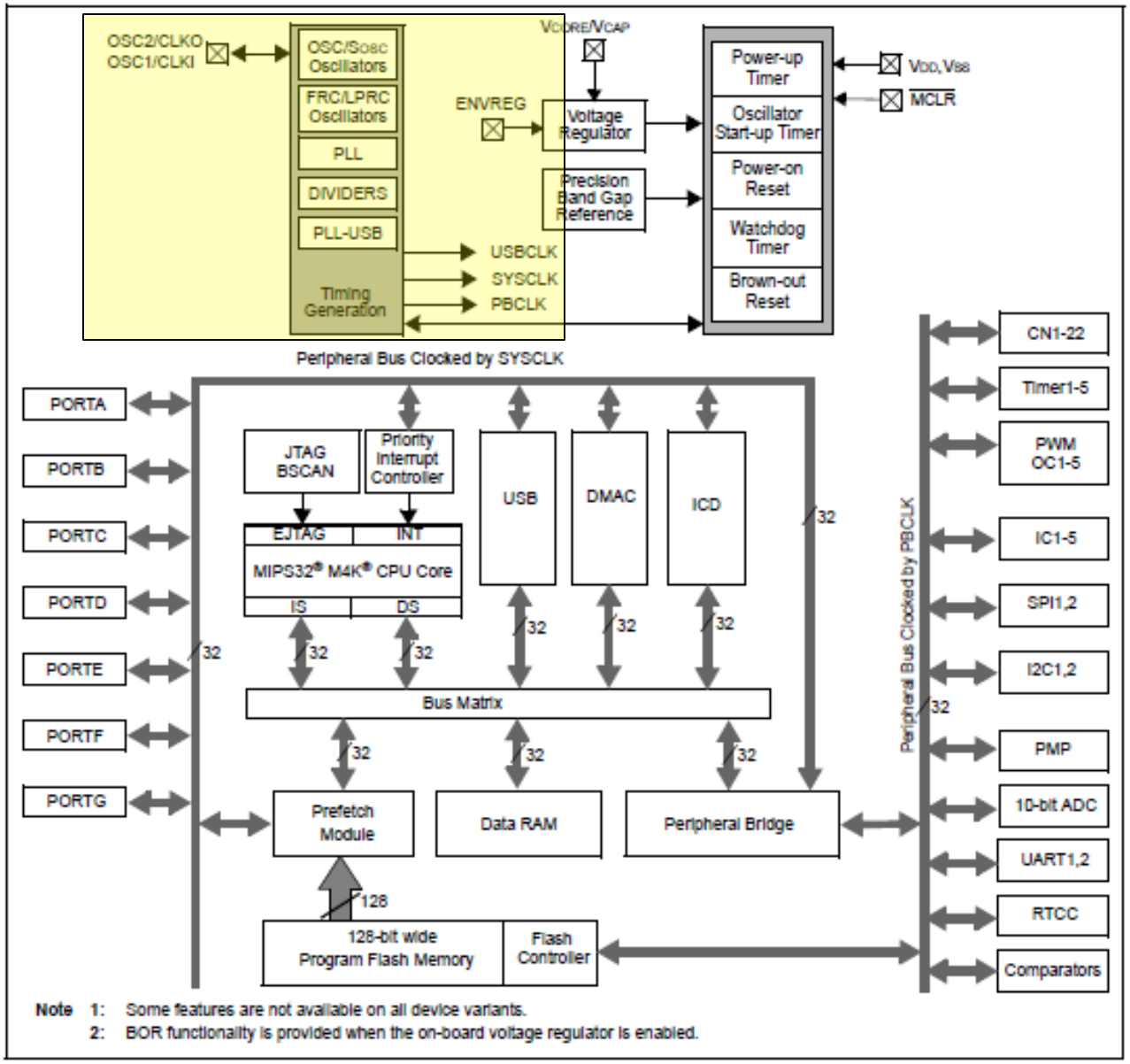
Configuring the Microcontroller Clocks

Dr. Edward Nava
ejnava@unm.edu

PIC Architecture

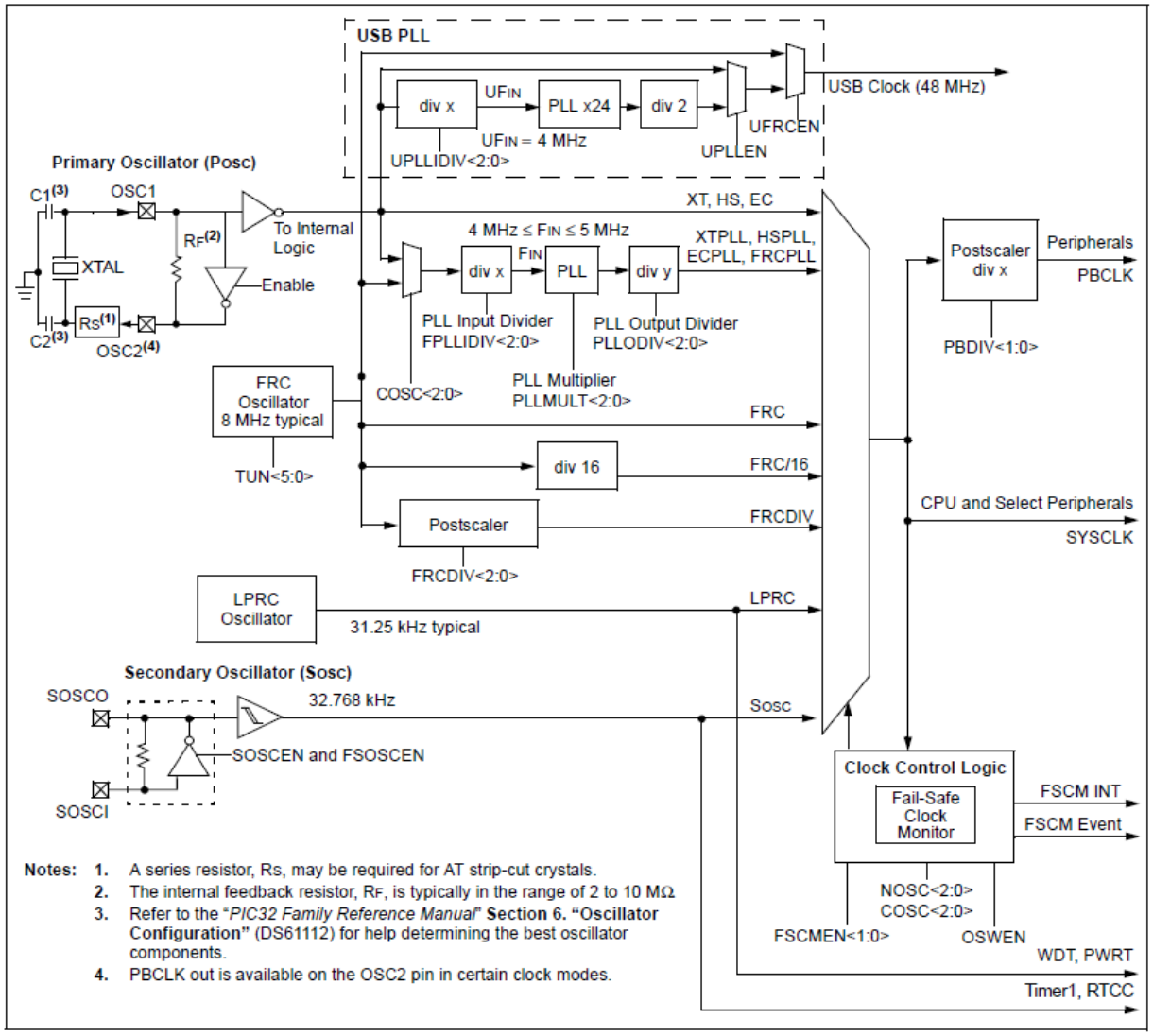


MX7 Board Clock Configuration

- The MX7 Board is equipped with an 8 MHz silicon resonator.
- We will use this resonator as our clock source and it connects to the POSC inputs.
- For proper impedance matching, we will operate in the EC mode.

PIC32 Microcontroller Clock

FIGURE 8-1: PIC32MX3XX/4XX FAMILY CLOCK DIAGRAM



Notes:
1. A series resistor, Rs, may be required for AT strip-cut crystals.
2. The internal feedback resistor, RF, is typically in the range of 2 to 10 MΩ
3. Refer to the "*PIC32 Family Reference Manual*" **Section 6. "Oscillator Configuration"** (DS61112) for help determining the best oscillator components.
4. PBCLK out is available on the OSC2 pin in certain clock modes.

Clock Configuration

- Configuration of the clock system is done using Special Function Registers (SFRs) in the PIC32 microcontroller.
- Configuration is done by setting appropriate bits at the corresponding SFR address.

Key References

- PIC Data Sheet – Table 8-1
- PIC32 Reference Manual – Section 6
- chipKIT MX7 Reference Manual

Clock Configuration SFRs

- The clock system is configured with following SFRs:

Name	Address
OSCCON	0xBF80F000
OSCTUN	0xBF80F010

- To change clock configuration during run-time, the following SFRs would be used.

Name	Address
DEVCFG1	0xBFC02FF8
DEVCFG2	0xBFC02FF4

Configuring Clock SFRs

TABLE 8-1: OSCILLATOR REGISTER MAP

Virtual Address (BF80_#)	Register Name(1)	Bit Range	Bits 31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0	All Resets(2)
F000	OSCCON	31:16	—	—	PLLODIV<2:0>			FRCDIV<2:0>			—	SOSCRDY	—	PBDIV<1:0>		PLLMULT<2:0>			0000
		15:0	—	COSC<2:0>			—	NOSC<2:0>			CLKLOCK	ULOCK	SLOCK	SLPEN	CF	UFRCEN	SOSCEN	OSWEN	0000
F010	OSCTUN	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	TUN<5:0>						0000

Legend: x = unknown value on Reset; — = unimplemented, read as '0'. Reset values are shown in hexadecimal.

Note 1: All registers in this table have corresponding CLR, SET and INV registers at their virtual addresses, plus offsets of 0x4, 0x8 and 0xC, respectively. See Section 12.1.1 "CLR, SET and INV Registers" for more information.

2: Reset values are dependent on the DEVCFGx Configuration bits and the type of Reset.

TABLE 29-1: DEVCFG: DEVICE CONFIGURATION WORD SUMMARY

Virtual Address (BFC0_#)	Register Name	Bit Range	Bits 31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0	All Resets
2FF0	DEVCFG3	31:16	FVBUSONIO	FUSBIDIO	—	—	—	FCANIO	FETHIO	FMIIEN	—	—	—	—	—	FSRSSEL<2:0>			xxxx
		15:0	USERID<15:0>																xxxx
2FF4	DEVCFG2	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	FPLLODIV<2:0>			xxxx
		15:0	UPLLEN	—	—	—	—	UPLLIDIV<2:0>			—	FPLLMUL<2:0>			—	FPLLIDIV<2:0>			xxxx
2FF8	DEVCFG1	31:16	—	—	—	—	—	—	—	—	FWDTEN	—	—	WDTPS<4:0>					xxxx
		15:0	FCKSM<1:0>		FPBDIV<1:0>		—	OSCIOFNC	POSCMOD<1:0>		IESO	—	FSOSCEN	—	—	FNOSC<2:0>			xxxx
2FFC	DEVCFG0	31:16	—	—	—	CP	—	—	—	BWP	—	—	—	—	PWP<7:4>				xxxx
		15:0	PWP<3:0>				—	—	—	—	—	—	—	—	ICESEL	—	DEBUG<1:0>		xxxx

Legend: x = unknown value on Reset; — = unimplemented, read as '0'. Reset values are shown in hexadecimal.

Clock Configuration SFRs

- The following SFR fields must be configured:
  - bit 29-27 **PLLODIV<2:0>:** Output Divider for PLL
  - bit 26-24 **FRCDIV<2:0>:** Internal Fast RC (FRC) Oscillator Clock Divider bits
  - bit 20-19 **PBDIV<1:0>:** Peripheral Bus Clock (PBCLK) Divisor bits
  - bit 18-16 **PLLMULT<2:0>:** Phase-Locked Loop (PLL) Multiplier bits
  - bit 14-12 **COSC<2:0>:** Current Oscillator Selection bits *
  - bit 10-8 **NOSC<2:0>:** New Oscillator Selection bits *

* Set via the FNOSC bit field in the DEVCFG1 SFR

SFR Field Settings

- Using preprocessor settings, we will use the following configurations:
  - FNOSC = PRIPLL
  - POSCMOD = EC
  - FPLLIDIV = DIV_2
  - FPLLMUL = MUL_20
  - FPLLODIV = DIV_1
  - FPBDIV = DIV_8

Setting Clock Configuration SFRs

- Clock configuration can be done by writing a 32 bit value to the appropriate SFR.
- The value to be written will have the individual bit fields set to the values corresponding to the desired options.
- In assembly language, this is done with a store word instruction (sw).
- When we write our programs using the c programming language, we will use a *#pragma* directive that causes the preprocessor to perform some implementation-defined action.

The #pragma directive

- *#pragma config* _____ - sets options defined by keyword templates.

Oscillator Configuration with #pragma directives

```
#pragma config FNOSC = PRIPLL       //designates primary oscillator
                                    // with PLL module
#pragma config POSCMOD = EC         // designate primary oscillator
                                    // input configuration
#pragma config FPLLIDIV = DIV_2     // config PLL input divider
                                    // to divide by 2
#pragma config FPLLMUL = MUL_20     // config PLL multiplier to
                                    // multiply by 20
#pragma config FPLLODIV = DIV_1     // config PLL output divider to
                                    // divide by 1
#pragma config FPBDIV = DIV_8       // config divider for peripheral
                                    // bus clock to divide by 8
```